Journal de développement

Choix faits durant le développement

TP 1

Afin de faciliter le développement de notre moteur de jeu, nous avons décidé d'utiliser l'IDE Clion qui appartient à la suite de logiciel Jetbrains.
Cet IDE dispose de plusieurs fonctionnalités utiles pour le développement comme la génération de code ou l'auto-complétion. De plus, l'ensemble des membres de notre groupe a eu l'habitude d'utiliser Clion pour les projets de C++ effectués au cours de notre formation.

Nous avons ensuite choisi GoogleTest comme framework de test pour notre moteur de jeu. L'utilisation d'un framework de test nous permet de pouvoir faire des tests de manière rapide et efficace et GoogleTest est l'un des plus utilisés et des plus pratiques.

Pour l'affichage graphique des résultats des calculs du moteur de jeu, nous avons choisi d'utiliser FreeGlut, étant une version libre et plus récente de Glut. Cependant, au cours de l'implémentation, il nous est apparu que FreeGlut utilise une version dépassée d'OpenGL, nous allons donc sans doute passer sur GLFW pour les prochaines itérations.

TP 2

Dans un premier temps, nous avons décidé de chercher l'ensemble des collisions dans la méthode AddContact. Puis au fur et à mesure du développement, il est apparu qu'il était plus simple que AddContact soit juste une fonction capable de repérer un contact entre 2 particules. Cette modification étant d'autant plus pertinente que la fonction AddContact était aussi nécessaire dans la classe ParticleCable et ParticleRod.

Pour faciliter la lisibilité du main, nous avons décidé de rajouter une classe WorldPhysics qui permettrait de gérer l'ensemble des générateurs de forces.

TP 3

Pour cette itération, peu de choses ont finalement été rajoutées en comparaison à l'itération précédente. Fort heureusement, dans cette itération, de nombreuses lignes de codes ont pu être réutilisées pour nous libérer du temps. Nous avons ainsi décidé, comme conseillé par notre professeur, de seulement modifier les classes existantes et de créer uniquement celles qui étaient nécessaires.

Comme pour l'itération précédente, nous avons gardé la classe World permettant une meilleure lisibilité du code. Nous avons aussi décidé de mettre un intégrateur dans chaque RigidBody et un intégrateur global dans la classe WorldPhysics qui appellerait l'ensemble des intégrateurs des RigidBody présents dans notre monde.

Après avoir décidé initialement de partir sur un rendu en 2 dimensions, nous avons finalement décidé de faire du rendu 3 dimensions pour permettre une meilleure visibilité des rotations et un plus joli rendu.

Difficultés rencontrées

TP 1

La première difficulté que notre équipe a rencontré est la maîtrise du langage C++. Bien qu'ayant fait quelques projets dans ce langage, s'y replonger après avoir fait du Java ou du python a nécessité quelques mises à jour. Clion nous a aussi posé quelques problèmes notamment l'installation de freeglut avec CMake. Il nous a fallu apprendre la grammaire de Cmake pour qu'il accepte la librairie FreeGlut.

De plus, rajouter GoogleTest nous a causé les mêmes problèmes, d'autant que cette librairie ne possédait pas de .lib et qu'il fallait donc compiler le code pour en obtenir un. Des mises à jour ont dû être faites sur le fichier en .lib pour qu'il accepte la configuration de nos différents PC.

Le point le plus difficile de cette itération fut cependant sans conteste l'utilisation de FreeGlut, OpenGL étant une technologie totalement inconnue pour nous. Réussir à faire déplacer un pixel d'un point à l'autre de l'écran a été plus compliqué que ce que nous pensions notamment dû au fait que la librairie FreeGlut est ancienne et que peu de documentation ont été trouvés sur le net concernant cette librairie.

TP 2

Plusieurs difficultés se sont posées tout au long de cette deuxième phase de développement. Tout d'abord, malgré le fait que, dans leur globalité, les fonctions est été implémentées juste après le cours, il a parfois été compliqué de comprendre la logique et la disposition des classes dû principalement à notre mécompréhension de ce qui était attendu pour cette phase.

Le fait d'avoir implémenté différents bouts de code tout au long de la deuxième partie a eu comme effet pervers de nous faire oublier l'utilité de certaines classe ou méthodes dû à la grande séparation entre le temps où elle ont été implémentées et le temps où elles nous ont vraiment servi. Ainsi, il a parfois été possible que certains attributs ou que certaines fonctions soient implémentées avant que l'on ne se rende compte que d'autres parties du code géraient déjà ces fonctionnalités.

De plus, cette implémentation par petits bouts ne pouvant être testés qu'à la toute fin, pas mal de bugs se sont finalement avérés être des erreurs assez simples à corriger mais difficiles à trouver dans l'ensemble des classes et des lignes de codes.

TP 3

Durant cette itération, quelques petits problèmes de compréhension se sont glissés et nous ont forcé à poser des questions à notre professeur ou à nos collègues. Le calcul de transformMatrix, par exemple, n'a pas bien été compris par notre groupe.

Notre groupe a eu comme à chaque itération des petits problèmes liés au C++ ou des bugs mineurs nous ayant ralentis dans le projet. On peut notamment parler de la mise en place de Matrix3, Matrix4 ou Quaternion qui nous ont obligé de changer certains opérateurs. Cela s'est cependant fait dans un temps relativement rapide.

Un bug majeur est survenu vers la fin du projet. Il générait des NaN à la moitié de l'exécution et supprimait l'affichage de notre boîte au milieu de son rendu. Ce dernier a été résolu lorsque nous avons rajouté la normalisation des quaternions, que nous avions oubliée auparavant.

Comme à chaque fois, FreeGlut nous a posé quelques problèmes quand à la mise en place des différentes scènes. La mise en place des scènes 2 dimensions, inspirés de celle faite dans l'itération suivante ont été simple à implémenter. Cependant, le choix de changer pour de la 3 dimensions plus tard dans le projet nous a apporté plusieurs problèmes non prévus originalement.

Astuce de programmation

TP 1

Pour plus de clarté, nous avons décidé de mettre un intégrateur dans la classe Particule afin de pouvoir y accéder plus facilement et nous avons mis en place un intégrateur global qui appellerait les intégrateurs de chaque particule.
Pour nous permettre de voir la particule évoluer dans un temps relativement long sans toucher aux paramètres initiaux de vitesse, d'accélération et de damping, nous avons décidé de faire reculer la caméra de FreeGlut pour que celle-ci nous affiche une plus longue distance.

TP 2

Afin de permettre une plus grande lisibilité dans le code, nous avons implémentés une classe worldPhysics dans laquelle les générateurs de forces seraient initiés. Cela permet de réduire le code contenu dans main.cpp et ainsi d'avoir un code plus clair.

Pour éviter tout problème de rendu, des tests ont préalablement été effectués avec OpenGL pour tester la création de plusieurs particules en même temps et la création de cercle. Nous réfléchissons d'ailleurs d'ors et déjà à comment faire le rendu du prochain TP qui utilisera des solides. D'autres logiciels de rendu autre que OpenGL sont examinés.

Malgré l'installation d'un plugin de test (GoogleTest), peu de tests ont été implémentés pour vérifier la bonne utilisation des classes. Dû à un manque de temps de notre part, il s'agit pourtant là d'une fonctionnalité qui nous aurait permis d'éviter la phase de débug durant les derniers jours avant le rendu.

TP 3

Comme nous l'avons signalé plus tôt, nous avons décidé·de réutiliser nos classes afin de nous éviter de réécrire certaines classes dont la structure est semblable à l'itération précédente.

Il est important de noter que Clion, bien qu'étant un formidable outil pour permettre de coder en C++ grâce notamment à l'auto-complétion, ne dispose pas d'un débogueur car encore dans une version relativement récente. Notre équipe a donc été obligé de faire des affichages à chaque fois qu'un bug ne faisant pas crasher le compilateur arrivait.

Pour faciliter les prochains débogage, nous utiliserons les ToString que nous avons implémenté dans cette itération pour afficher les différents objets du monde (Matrice, Vecteur, etc)